PAJ 07-01-02 02202886 JP **APPLICATION DEVELOPMENT SYSTEM AND ITS METHOD AND APPLICATION DEVELOPMENT PROGRAM AND APPLICATION GENERATION METHOD**

INVENTOR(S)- SEKI, TAKEO

PATENT APPLICATION NUMBER- 2001323903
DATE FILED- 2001-10-22
PUBLICATION NUMBER- 02202886 JP
DOCUMENT TYPE- A
PUBLICATION DATE- 2002-07-19
PATENT PRIORITY INFO- 2000328819, 2000-10-27, Japan
INTERNATIONAL PATENT CLASS- G06F00944
APPLICANT(S)- TOSHIBA CORP
PUBLICATION COUNTRY- Japan NDN- 043-0254-1229-0

PROBLEM TO BE SOLVED: To provide an application development system capable of easily developing an application whose maintainability is excellent capable of flexibly facilitating countermeasures to the change of the system environment of a platform or the like. SOLUTION: A designing tool 1 supports the design of an application based on the combination of a plurality of logical components based on logical component information 4, and outputs logical design information 5 acquired from the design. A source generation processing part 2 and a compiler 3 generates an application (performable file 9) performable on a specific platform based on the logical design information 5 outputted by the designing tool 1 and the physical mounting information (physical component information 6 and component library 8) of the software components. COPYRIGHT: (C)2002,JPO

NO-DESCRIPTORS

back to top

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（A）　(11)特許出願公開番号
特開2002－202886
（P2002－202886A）
(43)公開日　平成14年７月19日(2002.7.19)

(51)Int.Cl.'	識別記号	ＦＩ	テーマコード'(参考)
Ｇ０６Ｆ　9/44		Ｇ０６Ｆ　9/06	６２０Ａ　５Ｂ０７６

審査請求　未請求　請求項の数12　ＯＬ　（全 20 頁）

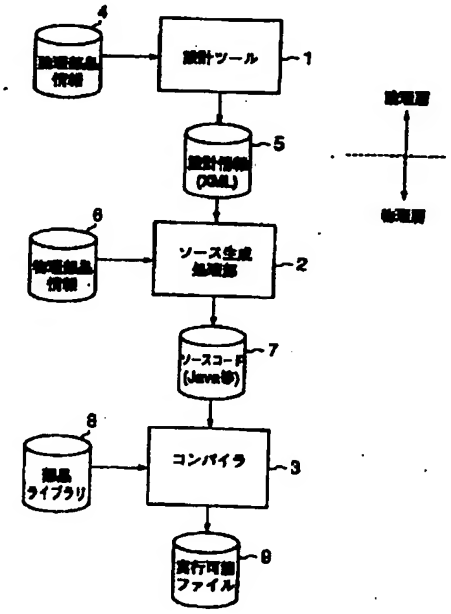
(21)出願番号　特願2001－323903(P2001－323903)
(22)出願日　平成13年10月22日(2001.10.22)
(31)優先権主張番号　特願2000－328819(P2000－328819)
(32)優先日　平成12年10月27日(2000.10.27)
(33)優先権主張国　日本（ＪＰ）

(71)出願人　000003078
株式会社東芝
東京都港区芝浦一丁目１番１号
(72)発明者　関　武　夫
東京都府中市東芝町１番地　株式会社東芝
府中事業所内
(74)代理人　100075812
弁理士　吉武　賢次　（外６名）

Ｆターム(参考)　5B076 AB15 DA06 DB01 DB04 DD05

(54)【発明の名称】　アプリケーション開発システム、その方法およびアプリケーション開発プログラムおよびアプリケーション生成方法

(57)【要約】
【課題】　プラットフォーム等のシステム環境の変化に柔軟に対応できる保守性に優れたアプリケーションの開発を容易に行うことができる、アプリケーション開発システムを提供する。
【解決手段】　設計ツール１は、論理部品情報４に基づいて複数の論理部品の組み合わせによるアプリケーションの設計を支援し、当該設計により得られた論理的設計情報５を出力する。ソース生成処理部２およびコンパイラ３は、設計ツール１により出力された論理的設計情報５とソフトウェア部品の物理的実装情報（物理部品情報６および部品ライブラリ８）とに基づいて特定のプラットフォーム上で実行可能なアプリケーション（実行可能ファイル９）を生成する。



【特許請求の範囲】

【請求項１】複数のソフトウェア部品を組み合わせてアプリケーションの開発を行うアプリケーション開発システムにおいて、

複数のソフトウェア部品のそれぞれに対応して設けられた論理部品であって当該ソフトウェア部品のうちシステム環境に依存しない部分を抽出してなる論理部品の定義情報を含む論理部品情報に基づいて、複数の論理部品の組み合わせによるアプリケーションの設計を支援し、当該設計により得られた論理的設計情報を出力する論理設計部と、

前記論理設計部により出力された論理的設計情報とソフトウェア部品の物理的実装情報とに基づいてシステム環境上で実行可能なアプリケーションを生成する物理実装部とを備えたことを特徴とするアプリケーション開発システム。

【請求項２】前記論理設計部は、前記論理部品情報に基づいて前記各論理部品の仕様情報を生成する仕様情報生成部と、前記仕様情報生成部により生成された前記各論理部品の仕様情報に基づいて前記各論理部品の組み合わせによるアプリケーションの設計を支援する設計部本体とを有することを特徴とする請求項１記載のアプリケーション開発システム。

【請求項３】前記各論理部品の仕様情報は、前記各論理部品の入出力仕様と前記各論理部品間の接続仕様と前記各論理部品の属性仕様とを含むことを特徴とする請求項２記載のアプリケーション開発システム。

【請求項４】前記物理実装部は、前記論理設計部により出力された論理的設計情報と、システム環境向けに実装された物理部品と論理部品との間の対応関係を含む物理部品情報とに基づいて、ソースコードを生成するソース生成処理部と、前記ソース生成処理部により生成されたソースコードと前記各物理部品の実装情報が格納された部品ライブラリとに基づいてシステム環境上で実行可能な実行可能ファイルを生成する実行可能ファイル生成部とを有することを特徴とする請求項１記載のアプリケーション開発システム。

【請求項５】前記物理実装部は、システム環境向けに実装された物理部品と論理部品との間の対応関係を含む物理部品情報に基づいて、ソースコードを生成するソース生成処理部と、前記ソース生成処理部により生成されたソースコードと前記各物理部品の実装情報が格納された部品ライブラリとに基づいてシステム環境上で実行可能な実行可能ファイルを生成する実行可能ファイル生成部と、実行可能ファイル生成部により生成された実行可能ファイルの実行時に、前記論理設計部により出力された論理的設計情報と、前記物理部品情報とに基づいて、部品オブジェクトを生成して、部品の属性および部品間の接続関係を設定する部品オブジェクト生成処理部とを有することを特徴とする請求項１記載のアプリケーション開発システム。

【請求項６】前記論理的設計情報は、前記各論理部品の属性情報と前記各論理部品間の接続情報とを含むことを特徴とする請求項１乃至５のいずれか記載のアプリケーション開発システム。

【請求項７】前記論理的設計情報はＸＭＬ言語により記述されていることを特徴とする請求項１乃至６のいずれか記載のアプリケーション開発システム。

【請求項８】前記論理設計部および前記物理実装部はネットワークを介して互いに接続されたクライアントコンピュータ上およびサーバコンピュータ上にそれぞれ設けられ、前記クライアントコンピュータと前記サーバコンピュータとの間で前記論理的設計情報が受け渡されることを特徴とする請求項１乃至７のいずれか記載のアプリケーション開発システム。

【請求項９】複数のソフトウェア部品を組み合わせてアプリケーションの開発を行うアプリケーション開発方法において、

複数のソフトウェア部品のそれぞれに対応して設けられた論理部品であって当該ソフトウェア部品のうちシステム環境に依存しない部分を抽出してなる論理部品の定義情報を含む論理部品情報に基づいて、複数の論理部品の組み合わせによるアプリケーションの設計を支援するステップと、

前記設計により得られた論理的設計情報とソフトウェア部品の物理的実装情報とに基づいてシステム環境上で実行可能なアプリケーションを生成するステップとを含むことを特徴とするアプリケーション開発方法。

【請求項１０】複数のソフトウェア部品を組み合わせてアプリケーションの開発を行うアプリケーション開発プログラムにおいて、

複数のソフトウェア部品のそれぞれに対応して設けられた論理部品であって当該ソフトウェア部品のうちシステム環境に依存しない部分を抽出してなる論理部品の定義情報を含む論理部品情報に基づいて、複数の論理部品の組み合わせによるアプリケーションの設計を支援させる手順と、

前記設計により得られた論理的設計情報を出力させる手順とをコンピュータに対して実行させることを特徴とするアプリケーション開発プログラム。

【請求項１１】複数のソフトウェア部品を組み合わせてアプリケーションの開発を行うアプリケーション開発プログラムにおいて、

複数のソフトウェア部品のそれぞれに対応して設けられた論理部品であって当該ソフトウェア部品のうちシステム環境に依存しない部分を抽出してなる論理部品の組み合わせにより得られた論理的設計情報を読み取らせる手順と、

この読み取られた論理的設計情報とソフトウェア部品の物理的実装情報とに基づいてシステム環境上で実行可能

なアプリケーションを生成させる手順とをコンピュータに対して実行させることを特徴とするアプリケーション開発プログラム。

【請求項１２】複数のソフトウェア部品を組み合わせてアプリケーションを生成するアプリケーション生成方法において、

複数のソフトウェア部品のそれぞれに対応して設けられた論理部品の特定情報を表示し、ユーザーに対して、論理部品の選択を促すステップと、

ユーザーから選択された論理部品について、その定義情報を含む論理部品情報に基づいて論理的設計情報を生成するステップと、

前記論理的設計情報とソフトウェア部品の物理的実装情報とに基づいてアプリケーションを生成するステップとを含むことを特徴とするアプリケーション生成方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、複数のソフトウェア部品を組み合わせてアプリケーションの開発を行うアプリケーション開発方法に関する。

【０００２】

【従来の技術】従来から、複数のソフトウェア部品を組み合わせてアプリケーションの開発を行うアプリケーション開発方法が知られている。このような従来のアプリケーション開発方法においては、図１９に示すように、特定のプラットフォーム４１向けに実装されたソフトウェア部品である物理部品４２を物理的レベルで組み合わせることにより（符号４３参照）、アプリケーション４０の開発を行っている。

【０００３】

【発明が解決しようとする課題】しかしながら、上述した従来のアプリケーション開発方法では、ソフトウェア部品の組み合わせが物理的レベルで記述されることとなるので、ソフトウェア部品が実装されるプラットフォーム（ＯＳやミドルウェア、言語、通信等のシステム環境）に依存した記述が必要となり、開発されたアプリケーション４０を他のプラットフォームへ移行することが困難となるという問題がある。

【０００４】また、上述した従来のアプリケーション開発方法では、開発されるアプリケーション４０においてアプリケーション本来の処理（例えば業務処理）の記述とプラットフォーム固有の処理の記述とが混在することとなるので、アプリケーション４０の保守性が悪いという問題がある。

【０００５】本発明はこのような点を考慮してなされたものであり、プラットフォーム等のシステム環境の変化に柔軟に対応できる保守性に優れたアプリケーションの開発を容易に行うことができる、アプリケーション開発システム、その方法およびアプリケーション開発プログラムおよびアプリケーション生成方法を提供することを目的とする。

【０００６】

【課題を解決するための手段】図２０は本発明によるアプリケーション開発方法を模式的に説明するための図である。図２０に示すように、本発明では、複数のソフトウェア部品のそれぞれに対応して、プラットフォーム４１に依存しない部分を抽出してなる論理部品５２を設け、これら各論理部品５２を組み合わせることにより（符号５３参照）、アプリケーションの設計を行う。そして、このような設計により得られた論理的設計情報とソフトウェア部品の物理的実装情報（物理部品４２と論理部品５２との間の対応関係を含む物理部品情報や、物理部品４２の実装情報が格納された部品ライブラリ）とに基づいて、特定のプラットフォーム４１上で実行可能なアプリケーション４０を生成する。

【０００７】このような基本原理の下で、本発明は、その第１の解決手段として、複数のソフトウェア部品を組み合わせてアプリケーションの開発を行うアプリケーション開発システムにおいて、複数のソフトウェア部品のそれぞれに対応して設けられた論理部品であって当該ソフトウェア部品のうちシステム環境に依存しない部分を抽出してなる論理部品の定義情報を含む論理部品情報に基づいて、複数の論理部品の組み合わせによるアプリケーションの設計を支援し、当該設計により得られた論理的設計情報を出力する論理設計部と、前記論理設計部により出力された論理的設計情報とソフトウェア部品の物理的実装情報とに基づいてシステム環境上で実行可能なアプリケーションを生成する物理実装部とを備えたことを特徴とするアプリケーション開発システムを提供する。

【０００８】なお、上述した第１の解決手段においては、前記論理的設計情報はＸＭＬ言語により記述されていることが好ましい。

【０００９】また、上述した第１の解決手段においては、前記論理設計部および前記物理実装部はネットワークを介して互いに接続されたクライアントコンピュータ上およびサーバコンピュータ上にそれぞれ設けられ、前記クライアントコンピュータと前記サーバコンピュータとの間で前記論理的設計情報が受け渡されることが好ましい。

【００１０】また、本発明は、その第２の解決手段として、複数のソフトウェア部品を組み合わせてアプリケーションの開発を行うアプリケーション開発方法において、複数のソフトウェア部品のそれぞれに対応して設けられた論理部品であって当該ソフトウェア部品のうちシステム環境に依存しない部分を抽出してなる論理部品の定義情報を含む論理部品情報に基づいて、複数の論理部品の組み合わせによるアプリケーションの設計を支援するステップと、前記設計により得られた論理的設計情報とソフトウェア部品の物理的実装情報とに基づいてシス

テム環境上で実行可能なアプリケーションを生成するステップとを含むことを特徴とするアプリケーション開発方法を提供する。

【0011】さらに、本発明は、その第3の解決手段として、複数のソフトウェア部品を組み合わせてアプリケーションの開発を行うアプリケーション開発プログラムにおいて、複数のソフトウェア部品のそれぞれに対応して設けられた論理部品であって当該ソフトウェア部品のうちシステム環境に依存しない部分を抽出してなる論理部品の定義情報を含む論理部品情報に基づいて、複数の論理部品の組み合わせによるアプリケーションの設計を支援させる手順と、前記設計により得られた論理的設計情報を出力させる手順とをコンピュータに対して実行させることを特徴とするアプリケーション開発プログラムを提供する。

【0012】さらにまた、本発明は、その第4の解決手段として、複数のソフトウェア部品を組み合わせてアプリケーションの開発を行うアプリケーション開発プログラムにおいて、複数のソフトウェア部品のそれぞれに対応して設けられた論理部品であって当該ソフトウェア部品のうちシステム環境に依存しない部分を抽出してなる論理部品の組み合わせにより得られた論理的設計情報を読み取らせる手順と、この読み取られた論理的設計情報とソフトウェア部品の物理的実装情報とに基づいてシステム環境上で実行可能なアプリケーションを生成させる手順とをコンピュータに対して実行させることを特徴とするアプリケーション開発プログラムを提供する。

【0013】なお、本発明は、その第5の解決手段として、複数のソフトウェア部品を組み合わせてアプリケーションを生成するアプリケーション生成方法において、複数のソフトウェア部品のそれぞれに対応して設けられた論理部品の特定情報を表示し、ユーザーに対して、論理部品の選択を促すステップと、ユーザーから選択された論理部品について、その定義情報を含む論理部品情報に基づいて論理的設計情報を生成するステップと、前記論理的設計情報とソフトウェア部品の物理的実装情報とに基づいてアプリケーションを生成するステップとを含むことを特徴とするアプリケーション生成方法を提供する。

【0014】本発明によれば、ソフトウェア部品の組み合わせにより行われるアプリケーションの開発を、(1)ソフトウェア部品のうちプラットフォーム等のシステム環境に依存しない部分を抽出してなる論理部品を組み合わせてアプリケーションを設計する処理（論理層）と、(2)この設計により得られた論理的設計情報に基づいて、システム環境に依存したソフトウェア部品である物理部品を組み合わせてアプリケーションを生成する処理（物理層）とに分割して行うので、論理層の各部については、最終的なシステム環境にかかわらず共通に利用することが可能となり、物理層の各部を最終的なシステム環境に応じて個別に用意することにより、システム環境に対応したアプリケーションの開発を行うことができる。このため、システム環境の変化に柔軟に対応できる保守性に優れたアプリケーションの開発を容易に行うことができる。

【0015】また、本発明によれば、論理層の処理と物理層の処理とを結び付ける論理的設計情報をXML言語という汎用性の高い言語で記述することにより、物理層および論理層の双方に自由度を与えることができる。このため、それぞれの層の差し替えを容易にし、しかも、アプリケーションの業務処理の論理表現の寿命を長くすることができる。

【0016】さらに、本発明によれば、論理層（論理設計部）および物理層（物理実装部）をクライアントコンピュータ上およびサーバコンピュータ上にそれぞれ設け、クライアントコンピュータとサーバコンピュータとの間で論理的設計情報を受け渡すようにすることにより、クライアントコンピュータとサーバコンピュータとの間での通信量を抑えることができる。このため、インターネット環境でも、クライアントコンピュータ側からサーバコンピュータ側のアプリケーションを開発することが可能となり、インターネット環境でのオープンなアプリケーション開発環境を提供するようなサービスに応用することが可能となる。

【0017】

【発明の実施の形態】以下、図面を参照して本発明の実施の形態について説明する。図1乃至図11は本発明によるアプリケーション開発システムの一実施の形態を説明するための図である。なお、本実施の形態では、アプリケーション開発システムを単一のコンピュータ上で実現する場合を例に挙げて説明する。

【0018】図1に示すように、本実施の形態に係るアプリケーション開発システムは、複数のソフトウェア部品を組み合わせてアプリケーションの開発を行うものであり、設計ツール（論理設計部）1と、ソース生成処理部2およびコンパイラ（実行可能ファイル生成部）3とを備えている。なお、ソース生成処理部2およびコンパイラ3により物理実装部が構成されている。

【0019】このうち、設計ツール1は、論理部品情報4に基づいて複数の論理部品の組み合わせによるアプリケーションの設計を支援し、当該設計により得られた論理的設計情報5を出力するものである。ここで、設計ツール1に入力される論理部品情報4は、論理部品の定義情報を含むものであり、例えばXML言語により文字列データとして記述されている（図7(a)(b)参照）。なお、論理部品とは、アプリケーションの設計のために用いられる部品であり、複数のソフトウェア部品のそれぞれに対応して設けられ、ソフトウェア部品のうちプラットフォーム（システム環境）に依存しない部分を抽出したものである。また、設計ツール1により出力される論

理的設計情報5は、各論理部品の属性情報と各論理部品間の接続情報とを含むものであり、例えばXML言語により文字列データとして記述されている（図8参照）。

【0020】一方、ソース生成処理部2およびコンパイラ3は、設計ツール1により出力された論理的設計情報5とソフトウェア部品の物理的実装情報（物理部品情報6および部品ライブラリ8）とに基づいて特定のプラットフォーム上で実行可能なアプリケーション（実行可能ファイル9）を生成するものである。

【0021】具体的には、ソース生成処理部2は、設計ツール1により出力された論理的設計情報5と物理部品情報6とに基づいてソースコード7を生成するものである。なお、ソース生成処理部2に入力される物理部品情報6は、特定のプラットフォーム向けに実装された物理部品と論理部品との間の対応関係を含むものであり、例えばXML言語により文字列データとして記述されている（図9および図10(a)(b)(c)参照）。なお、論理部品と物理部品との対応関係は1対1に限らず、1対多でもよい。例えば、論理部品“A”に対して物理部品“a”となる場合もあれば、論理部品“A”に対して物理部品“a＋b”となる場合もあり、同じ論理部品“A”に対しても複数の物理部品を定義できる。さらに、より具体的には、論理部品“チェック付き更新”に対して物理部品が“UpdateWithCheck”となる場合もあれば、論理部品“チェック付き更新”に対して物理部品が“Check”＋“Update”となる場合もあり、同じ論理部品“チェック付き更新”に対しても複数の物理部品を定義できる。また、ソース生成処理部2により出力されるソースコード7は、例えばＪａｖａ（登録商標）等のプログラミング言語により文字列データとして記述されている（図11参照）。

【0022】また、コンパイラ3は、ソース生成処理部2により生成されたソースコード7と各物理部品の実装情報（インタフェースや型定義等）が格納された部品ライブラリ8とに基づいて特定のプラットフォーム上で実行可能な実行可能ファイル9を生成するものである。なお、部品ライブラリ8に格納される物理部品としては、ＥＪＢ、ＪａｖａＢｅａｎｓおよびＪａｖａクラス等が用いられる。

【0023】以上において、設計ツール1、論理部品情報2および論理的設計情報5により論理層が構成され、ソース生成処理部2、コンパイラ3、物理部品情報6、ソースコード7、部品ライブラリ8および実行可能ファイル9により物理層が構成されている。

【0024】図2は図1に示す設計ツール1の詳細を示す図である。

【0025】図2に示すように、設計ツール1は、論理部品情報4に基づいて論理部品ツール情報12を生成する論理部品ツール情報設定部（仕様情報生成部）11と、論理部品ツール情報設定部11により生成された論理部品ツール情報12に基づいて各論理部品の組み合わせによるアプリケーションの設計を支援する設計ツール本体（設計部本体）13とを有している。なお、論理部品情報4は、論理部品の定義情報として、論理部品基本情報4a（個々の論理部品のＩＤや属性情報、入出力情報等）と、論理部品関連情報4b（個々の論理部品間の接続情報）とを含んでいる。また、論理部品ツール情報12は、各論理部品の仕様情報を表すものであり、各論理部品の入出力仕様（論理部品入出力仕様12a）と、各論理部品間の接続仕様（論理部品接続仕様12b）と、各論理部品の属性仕様（論理部品属性仕様12c）とを含んでいる。なお、設計ツール本体13から出力される論理的設計情報5は、各論理部品の属性情報（部品属性情報5a）と、各論理部品間の接続情報（部品接続情報5b）とを含んでいる。

【0026】また、設計ツール1は、設計ツール本体13に接続された画面入出力インタフェース部14を有している。ここで、画面入出力インタフェース部14は、設計の基礎となる複数の論理部品をツール画面上でアイコンとして表現するグラフィカルユーザインタフェース（ＧＵＩ）を提供するものであり、アイコン間を関係線で結んだグラフ構造により、開発対象となるアプリケーションを構成するソフトウェア部品の組み合わせを表現することができるようになっている。なお、画面入出力インタフェース部14は、Ｗｅｂブラウザ等のアプリケーション上で動作するインタフェースプログラムとして実現することが可能である。

【0027】図3はツール画面の一例を示す図である。図3に示すように、ツール画面20は、パレット領域21と、作業領域22とを有している。このうち、パレット領域21には、論理部品情報4に基づいて、設計の基礎となる論理部品に対応する複数のアイコン23が一覧表示されている。また、作業領域22には、パレット領域21に配置されたアイコン23に基づいて貼り付けられた複数のアイコン24が表示されている。なお、作業領域22に配置されたアイコン24は、開発対象となるアプリケーションを構成する個々のソフトウェア部品に対応している。

【0028】このようなツール画面20において、ユーザは、マウス等の入力装置によりアイコンに対して各種の操作（アイコンの貼り付け、移動および関係付け等）を行うことができるようになっており、これらのアイコンに対する操作に連動して論理部品の組み合わせによりアプリケーションの設計を行うことができるようになっている。

【0029】具体的には例えば、パレット領域21に配置された複数のアイコン23のうちの任意のアイコン（アイコンA，B，C）の上でマウスの左ボタンをクリックした後、作業領域22の任意の位置でマウスの左ボタンを再度クリックすることにより、作業領域22にア

イコン24（アイコンA，B，C）を貼り付けることができる（符号26参照）。なお、このようにして作業領域22にアイコン24が貼り付けられた後、アイコン24の上でマウスの右ボタンをクリックしてメニュー（図示せず）をポップアップし、このメニュー中の項目を適宜選択することにより、論理部品の属性等（論理部品の名前（"parent"，"child1"等）等）を設定することができる。

【0030】また、作業領域22に配置されたアイコンのうち、関係を設定する元のコンポーネントに対応するアイコン（図3では、アイコンA）の出力端子の上でマウスの左ボタンを押し、左ボタンを押した状態で、関係を設定する先のコンポーネントに対応するアイコン（図3では、アイコンB、C）までマウスを動かす（ドラッグする）ことにより、アイコン間の関係線25を定義することができる（符号27参照）。

【0031】なお、ツール画面20上でのアイコンに対する操作（アイコンの配置、移動および関係付け等）は、マウス等の基本操作により実現される。なお、マウス等の基本操作は、論理部品間で統一されている。

【0032】次に、このような構成からなる本実施の形態の作用について説明する。

【0033】まず、図4および図5により、図1および図2に示すアプリケーション開発システムの論理層（設計ツール1）の処理について説明する。

【0034】図4は図1および図2に示す設計ツール1の論理部品ツール情報設定部11の動作を説明するためのフローチャートである。

【0035】図4に示すように、設計ツール1の論理部品ツール情報設定部11は、まず、論理部品情報4のうち論理部品基本情報4aを読み取り（ステップ101）、この論理部品基本情報4aに基づいて、論理部品ごとに論理部品入出力仕様12aおよび論理部品属性仕様12cを生成する（ステップ102および103）。次に、論理部品情報4のうち論理部品関連情報4bを読み取り（ステップ104）、この論理部品関連情報4bに基づいて、論理部品の組み合わせごとに論理部品接続仕様12bを生成する（ステップ105）。

【0036】図7(a)(b)は論理部品情報4の一例を示す図である。このうち、図7(a)は、論理部品基本情報4aの一部（個々の論理部品のID）と論理部品関連情報4b（個々の論理部品間の接続情報）とを含む全体情報を示している。また、図7(b)は、論理部品基本情報4aの一部（個々の論理部品の属性情報および入出力情報）を含む個別情報を示している。

【0037】ここで、図7(a)の符号（A）の部分には、個々の論理部品のIDが記述されている（<;component type="A"/>;等）。また、符号（B）の部分には、論理部品接続仕様12bの元となる個々の論理部品間の接続情報が記述されている（<;connection from="A"to="B"/>;等）。なお、「<;connection from="A"to="B"/>;」の部分は、ソフトウェア部品Aからソフトウェア部品Bに接続できることを表している。

【0038】また、図7(b)の符号（C）の部分には、論理部品属性仕様12cの元となる個々の論理部品の属性情報が記述されている（<;property name="name" type="string"/>;等）。なお、「<;property name="name" type="string"/>; <;property name="option" type="string"/>;」の部分は、"name"と"option"という2つの文字列属性があることを表している。また、符号（D）の部分には、論理部品入出力仕様12aの元となる個々の論理部品の入出力情報が記述されている（<;terminalid="out" type="out" min=1 max=3>;...<;/terminal>;等）。なお、「<;terminal id="out" type="out" min=1 max=3>; <;connectwith type="B"/>; <;connectwith type="C"/>; <;/terminal>;」の部分は、"out"という出力端子があり、1本から3本の接続ができ、接続先はソフトウェア部品BまたはCであることを表している。

【0039】図5は図1および図2に示す設計ツール1の設計ツール本体13の動作を説明するためのフローチャートである。

【0040】図5に示すように、設計ツール1の画面入出力インタフェース部14は、設計ツール本体13による制御の下で、図3に示すツール画面20を呈示し、ユーザによるアイコンの操作を受け付けて、各論理部品の組み合わせによるアプリケーションの設計を支援する（ステップ201および202）。

【0041】具体的には、画面入出力インタフェース部14は、ユーザが、ツール画面20のパレット領域21に配置された複数のアイコン23のうちの任意のアイコン（アイコンA，B，C）の上でマウスの左ボタンをクリックした後、作業領域22の任意の位置でマウスの左ボタンを再度クリックすることにより、作業領域22にアイコン24（アイコンA，B，C）を貼り付ける（ステップ201）。

【0042】また、作業領域22に配置されたアイコンのうち、関係を設定する元のコンポーネントに対応するアイコン（図3では、アイコンA）の出力端子の上でマウスの左ボタンを押し、左ボタンを押した状態で、関係を設定する先のコンポーネントに対応するアイコン（図3では、アイコンB，C）までマウスを動かす（ドラッグする）ことにより、アイコン間の関係線25を定義する（ステップ202）。

【0043】その後、設計ツール本体13は、論理部品ツール情報12の論理部品入出力仕様12aに基づいて各論理部品の接続点の妥当性をチェックし（ステップ203）、妥当でない場合には、ステップ202に戻って処理を続け、妥当である場合には、ステップ205に進む（ステップ204）。

【0044】また、設計ツール本体13は、論理部品ツ

ール情報１２の論理部品接続仕様１２ｂに基づいて各論理部品間の接続関係を妥当性をチェックし（ステップ２０５）、妥当でない場合には、ステップ２０２に戻って処理を続け、妥当である場合には、ステップ２０７に進む（ステップ２０６）。

【００４５】その後、ユーザが、作業領域２２に貼り付けられたアイコン２４の上でマウスの右ボタンをクリックしてメニュー（図示せず）をポップアップし、このメニュー中の項目を適宜選択して論理部品の属性等を設定すると（ステップ２０７）、論理部品ツール情報１２の論理部品属性仕様１２ｃに基づいて各論理部品の属性の妥当性をチェックし（ステップ２０８）、妥当でない場合には、ステップ２０７に戻って処理を続け、妥当である場合には、ステップ２１０に進む（ステップ２０９）。

【００４６】最後に、全ての論理部品の処理が終了したか否かを判断し（ステップ２１０）、処理が終了している場合には、論理的設計情報５を出力する（ステップ２１１）。

【００４７】図８は論理的設計情報５の一例を示す図であり、論理的設計情報５は、部品属性情報５ａ（符号（Ｅ）参照）と部品接続情報５ｂ（符号（Ｆ）参照）とを含んでいる。ここで、図８に示す論理的設計情報５は、図３に示すような論理部品の組み合わせ（アイコンＡ（名前が"parent"で、オプションが"1"）の下にアイコンＢ（名前が"child1"で、オプションが"11"）とアイコンＣ（名前が"child2"で、オプションが"12"）とがツリー状に接続される組み合わせ）に対応している。

【００４８】次に、図６により、図１に示すアプリケーション開発システムの物理層（ソース生成処理部２およびコンパイラ３）の処理について説明する。

【００４９】図６は図１に示すソース生成処理部２の動作を説明するためのフローチャートである。

【００５０】図６に示すように、ソース生成処理部２は、設計ツール１により出力された論理的設計情報５（部品属性情報５ａおよび部品接続情報５ｂ）と物理部品情報６とに基づいてソースコード７を生成する。

【００５１】図９および図１０(a)(b)(c)は物理部品情報６の一例を示す図である。このうち、図９は、物理部品情報６のうちソフトウェア部品ごとのコード生成方法に関連する情報を示しており、特定のプラットフォーム向けに実装された物理部品と論理部品との間の対応関係や、個々の物理部品の属性関連情報および接続関連情報等を含んでいる。また、図１０(a)(b)(c)は、全体のコード生成方法に関連する情報を示している。

【００５２】ここで、図９の符号（Ｇ）の部分には、個々のソフトウェア部品の初期化コード生成方法が記述されている（<;new code="ClassX%name%=ClassX.newinstance()"/>;等）。また、符号（Ｈ）の部分には、個々のソフトウェア部品の属性設定コード生成方法が記述されている（<;property name="option" code="%name%.addItem("OPTION","%value%","string")"/>;等）。さらに、符号（Ｉ）の部分には、個々のソフトウェア部品の部品接続設定コード生成方法が記述されている（<;code="%from%.addItem("CHILD",%to%);/>;等）。なお、符号（Ｇ）（Ｈ）の部分および符号（Ｉ）の部分はそれぞれ、ソースコード７の属性設定コード７ｂおよび部品接続設定コード７ｃの元となる部分であり、「%xxx%」の項目が論理的設計情報５の値と置き換えられることによりソースコード７の該当部分が生成される。

【００５３】また、図１０(a)の符号（Ｊ）の部分には、ソースコード７の初期設定コード７ａの元となる初期設定コード生成方法が記述され（<;init file="fwlinit.java"/>;）、符号（Ｋ）の部分には、ソースコード７の終了処理設定コード７ｄの元となる終了処理設定コード生成方法が記述されている（<;term file="fwlterm.java"/>;）。なお、「<;init file="fwlinit.java"/>;」の部分は、図１０(b)に示すテンプレートファイル（fwlinit.java）をインポートするための記述であり、「<;term file="fwlterm.java"/>;」の部分は、図１０(c)に示すテンプレートファイル（fwlterm.java）をインポートするための記述である。

【００５４】このような物理部品情報６に基づいて、ソース生成処理部２は、図６に示すように、まず、ソースコード７の初期設定コード７ａを出力する（ステップ３０１）。

【００５５】次に、論理的設計情報５のうち部品属性情報５ａを読み取り（ステップ３０２）、この部品属性情報５ａと物理部品情報６とに基づいて、ソフトウェア部品ごとにソースコード７の属性設定コード７ｂを生成する（ステップ３０３）。

【００５６】その後、全てのソフトウェア部品の処理が終了したか否かを判断し（ステップ３０４）、処理が終了していない場合には、ステップ３０２に戻って処理を続け、処理が終了している場合には、ステップ３０５に進む。

【００５７】次に、論理的設計情報５のうち部品接続情報５ｂを読み取り（ステップ３０５）、この部品属性情報５ａと物理部品情報６とに基づいて、ソフトウェア部品ごとにソースコード７の部品接続設定コード７ｃを生成する（ステップ３０６）。

【００５８】その後、全てのソフトウェア部品の処理が終了したか否かを判断し（ステップ３０７）、処理が終了していない場合には、ステップ３０５に戻って処理を続け、処理が終了している場合には、物理部品情報６に基づいてソースコード７の終了処理設定コード７ｄを出力し（ステップ３０８）、全体の処理を終了する。図１１はこのようにして生成されるソースコード７の一例を示す図である。

【００５９】なお、このようにしてソース生成処理部２

によりソースコード７が生成された後、図１に示すように、ソースコード７をコンパイラ３にかけることにより、部品ライブラリ８に格納された各物理部品の実装情報に基づいて特定のプラットフォーム上で実行可能な実行可能ファイル９を生成する。

【００６０】このように本実施の形態によれば、ソフトウェア部品の組み合わせにより行われるアプリケーションの開発を、(1)ソフトウェア部品のうちプラットフォーム等のシステム環境に依存しない部分を抽出してなる論理部品を組み合わせてアプリケーションを設計する処理（論理層）と、(2)システム環境に依存したソフトウェア部品である物理部品を組み合わせてアプリケーションを生成する処理（物理層）とに分割して行うので、論理層の各部については、最終的なシステム環境にかかわらず共通に利用することが可能となり、物理層の各部を最終的なシステム環境に応じて個別に用意することにより、システム環境に対応したアプリケーションの開発を行うことができる。このため、システム環境の変化に柔軟に対応できる保守性に優れたアプリケーションの開発を容易に行うことができる。

【００６１】具体的には、（１）論理層（設計ツール１）の処理により得られた論理的設計情報５に基づいて、複数のシステム環境に対応したアプリケーションを物理層の部品および処理（ソース生成処理部２、物理部品情報６、部品ライブラリ８およびコンパイラ３等）を差し替えることにより生成することができる。また、（２）論理層の処理により得られた論理的設計情報５は、システム環境に影響されることが少ないので、実行するシステム環境が技術進歩等により大きく変わった場合でも、新しいシステム環境に合わせた物理層の部品および処理をアプリケーションごとに用意することにより、新しいシステム環境上で実行可能なアプリケーションを容易に生成することができる。さらに、（３）論理層の処理は、物理層に縛られることなく特定の論理的設計情報５を出力することに限定することができるので、複数のシステム環境上で動作する開発ツールを容易に準備することができる。

【００６２】また、本実施の形態によれば、論理層の処理と物理層の処理とを結び付ける論理的設計情報５をＸＭＬ言語という汎用性の高い言語で記述しているので、物理層および論理層の双方に自由度を与えることができ、それぞれの層の差し替えを容易にし、しかも、アプリケーションの業務処理の論理表現の寿命を長くすることができる。

【００６３】なお、上述した実施の形態においては、ソース生成処理部２により、属性設定コードおよび部品接続設定コードを含むソースコード７を生成するとともに、コンパイラ３により、このソースコード７に基づいてソフトウェア部品の属性およびソフトウェア部品間の接続関係が設定された実行可能ファイル９を生成するようにしているが、図１２乃至図１７に示す他の実施の形態のように、ソフトウェア部品の属性およびソフトウェア部品間の接続関係については、実行時に設定するようにしてもよい。

【００６４】以下、図１２乃至図１７により、本発明によるアプリケーション開発システムの他の実施の形態について説明する。なお、図１２乃至図１７に示す実施の形態は、実行可能ファイルの実行時にソフトウェア部品の属性およびソフトウェア部品間の接続情報を設定するようにした点を除いて、他は、図１乃至図１１に示す実施の形態と略同一である。図１２乃至図１７に示す実施の形態において、図１乃至図１１に示す実施の形態と同一部分には同一符号を付して詳細な説明は省略する。

【００６５】図１２に示すように、本発明の他の実施の形態においては、ソース生成処理部２′により、インタプリタ型部品オブジェクト生成コードを含むソースコード７を生成し、このソースコード７に基づいてコンパイラ３により生成された実行可能ファイル９の実行時に、実行処理部１５および部品オブジェクト生成処理部１６により、ソフトウェア部品の属性およびソフトウェア部品間の接続情報を設定する。

【００６６】図１３は図１２に示すソース生成処理部２′の動作を説明するためのフローチャートである。

【００６７】図１３に示すように、ソース生成処理部２′は、物理部品情報６に基づいてソースコード７を生成する。

【００６８】図１５および図１６(a)(b)(c)(d)は物理部品情報６の一例を示す図である。このうち、図１５は、物理部品情報６のうちソフトウェア部品ごとの部品オブジェクト生成方法に関連する情報を示している。また、図１６(a)(b)(c)(d)は、全体のコード生成方法に関連する情報を示している。

【００６９】図１６(a)の符号（Ｌ）の部分には、ソースコード７の初期設定コード７ａ（図１３参照）の元となる初期設定コード生成方法が記述され（<;init file="fwlinit.java"/>;）、符号（Ｍ）の部分には、ソースコード７のインタプリタ型部品オブジェクト生成コード１７ｅ（図１３参照）の元となる部品オブジェクト生成方法が記述され（<;main file="fwlmain.java"/>;）、符号（Ｎ）の部分には、ソースコード７の終了処理設定コード７ｄ（図１３参照）の元となる終了処理設定コード生成方法が記述されている（<;term file="fwlterm.java"/>;）。なお、「<;init file="fwlinit.java"/>;」の部分は、図１６(b)に示すテンプレートファイル（fwlinit.java）をインポートするための記述であり、「<;main file="fwlmain.java"/>;」の部分は、図１６(c)に示すテンプレートファイル（fwlmain.java）をインポートするための記述であり、「<;term file="fwlterm.java"/>;」の部分は、図１６(d)に示すテンプレートファイル（fwlterm.java）をインポートするための記述である。

【００７０】このような物理部品情報６に基づいて、ソース生成処理部２′は、図１３に示すように、まず、ソースコード７の初期設定コード７ａを出力する（ステップ４０１）。

【００７１】次に、ソースコード７のインタプリタ型部品オブジェクト生成コード７ｅを出力する（ステップ４０２）。

【００７２】その後、物理部品情報６に基づいてソースコード７の終了処理設定コード７ｄを出力し（ステップ４０３）、全体の処理を終了する。図１７はこのようにして生成されるソースコード７の一例を示す図である。

【００７３】なお、このようにしてソース生成処理部２′によりソースコード７が生成された後、図１２に示すように、ソースコード７をコンパイラ３にかけることにより、部品ライブラリ８に格納された各物理部品の実装情報に基づいて特定のプラットフォーム上で実行可能な実行可能ファイル９を生成する。

【００７４】このようにして生成された実行可能ファイル９は実行処理部１５により実行されるが、実行処理部１５はその処理の過程で部品オブジェクト生成処理部１６を呼び出し、これにより、ソフトウェア部品の部品オブジェクトを生成するとともに、当該ソフトウェア部品の属性およびソフトウェア部品間の接続情報を設定する。

【００７５】図１４は図１２に示す部品オブジェクト生成処理部１６の動作を説明するためのフローチャートである。

【００７６】図１４に示すように、部品オブジェクト生成処理部１６は、設計ツール１により出力された論理的設計情報５（部品属性情報５ａおよび部品接続情報５ｂ）と物理部品情報６とに基づいて、部品オブジェクトの生成、および属性および部品間の接続情報の設定等を行う。

【００７７】具体的には、まず、論理的設計情報５のうち部品属性情報５ａを読み取り（ステップ５０１）、この部品属性情報５ａと物理部品情報６とに基づいて、ソフトウェア部品ごとに部品オブジェクトを生成し、属性を設定する（ステップ５０２）。

【００７８】その後、全てのソフトウェア部品の処理が終了したか否かを判断し（ステップ５０３）、処理が終了していない場合には、ステップ５０１に戻って処理を続け、処理が終了している場合には、ステップ５０４に進む。

【００７９】次に、論理的設計情報５のうち部品接続情報５ｂを読み取り（ステップ５０４）、この部品属性情報５ａと物理部品情報６とに基づいて、ソフトウェア部品ごとに相互参照情報を設定し、部品間を接続する（ステップ５０５）。

【００８０】その後、全てのソフトウェア部品の処理が終了したか否かを判断し（ステップ５０６）、処理が終了していない場合には、ステップ５０４に戻って処理を続け、処理が終了している場合には、全体の処理を終了する。

【００８１】なお、上述した図１乃至図１１に示す実施の形態、および図１２乃至図１７に示す実施の形態においては、アプリケーション開発システムを単一のコンピュータ上で実現する場合を例に挙げて説明したが、これに限らず、アプリケーション開発システムを図１８に示すような分散開発環境で実現することも可能である。具体的には、図１８に示すように、インターネット等のネットワーク３２を介して互いに接続された複数のクライアントコンピュータ３１およびサーバコンピュータ３３において、論理層（設計ツール１および論理部品情報４）をクライアントコンピュータ３１上に設け、物理層（ソース生成処理部２、コンパイラ３、物理部品情報６、ソースコード７、部品ライブラリ８および実行可能ファイル９）をサーバコンピュータ３３上に設けるようにしてもよい。なお、この場合には、クライアントコンピュータ３１とサーバコンピュータ３３との間で、物理部品の実装情報を伴わないＸＭＬ言語等による論理的設計情報５が受け渡されることとなるので、通信量を抑えることができる。このため、インターネット環境でも、クライアントコンピュータ３１側からサーバコンピュータ３３側のアプリケーションを開発することが可能となり、インターネット環境でのオープンなアプリケーション開発環境を提供するようなサービスに応用することが可能となる。

【００８２】また、上述した図１乃至図１１に示す実施の形態、および図１２乃至図１７に示す実施の形態においては、コンパイラ３により、ソース生成処理部２により生成されたソースコード７と部品ライブラリ８とを統合しているが、これに限らず、ソースコード７と部品ライブラリ８とを部品選択手法や部品プラグイン手法等により統合するようにしてもよい。

【００８３】さらに、上述した図１乃至図１１に示す実施の形態、および図１２乃至図１７に示す実施の形態においては、論理部品情報４および物理部品情報６をＸＭＬ言語により記述しているが、これに限らず、表形式等の任意の形式で記述することができる。

【００８４】なお、上述した図１乃至図１１に示す実施の形態、および図１２乃至図１７に示す実施の形態において、アプリケーション開発システムの設計ツール１、ソース生成処理部２およびコンパイラ３はプログラムとして実現することができる。このようなプログラムからなるアプリケーション開発プログラムは、例えば図１８に示すような分散開発環境であれば、フレキシブルディスク３４やＣＤ－ＲＯＭ３５等のコンピュータ読み取り可能な記録媒体に記録することが可能であり、クライアントコンピュータ３１およびサーバコンピュータ３３上において、図４乃至図６に示すような手順および図１３

および図１４に示すような手順に従ってアプリケーションを開発することができる。

【００８５】なお、本発明における記録媒体としては、フレキシブルディスクやＣＤ－ＲＯＭに限られず、磁気ディスク、内部メモリ、ハードディスク、光ディスク（ＣＤ－Ｒ、ＤＶＤ（Digital Versatile Disk）等）、光磁気ディスク（ＭＯ（Magneto Optical）等）、半導体メモリ等のように、プログラムを記録でき、かつコンピュータが読み取り可能な記録媒体であれば、その記録形式はいずれかの形式であってもよい。また、記録媒体としては、ネットワーク上で伝送される際の搬送波や、情報伝達媒体も含む。

【００８６】また、記録媒体からコンピュータにインストールされたプログラムの指示に基づきコンピュータ上で稼働しているＯＳ（オペレーティングシステム）や、データベース管理ソフト、ネットワークソフト等のＭＷ（ミドルウェア）等が本発明を実現するための各処理の一部を実行してもよい。

【００８７】さらに、本発明における記録媒体は、コンピュータと独立した媒体に限らず、ＬＡＮやインターネット等により伝送されたプログラムをダウンロードして記憶または一時記憶した記録媒体も含まれる。

【００８８】さらにまた、記録媒体は１つに限らず、複数の媒体から本発明における処理が実行される場合も本発明における記録媒体に含まれ、媒体構成はいずれの構成であってもよい。

【００８９】

【発明の効果】以上説明したように本発明によれば、システム環境の変化に柔軟に対応できる保守性に優れたアプリケーションの開発を容易に行うことができる。

【図面の簡単な説明】

【図１】本発明によるアプリケーション開発システムの一実施の形態を示すシステム構成図。

【図２】図１に示すアプリケーション開発システムの設計ツール（論理設計部）の詳細を示すブロック図。

【図３】図１に示すアプリケーション開発システムにおいてユーザに対して呈示されるツール画面の一例を示す図。

【図４】図１および図２に示す設計ツールの論理部品ツール情報設定部（仕様情報生成部）の動作を説明するためのフローチャート。

【図５】図１および図２に示す設計ツールの設計ツール本体（設計部本体）の動作を説明するためのフローチャート。

【図６】図１に示すソース生成処理部の動作を説明するためのフローチャート。

【図７】図１に示すアプリケーション開発システムで用いられる論理部品情報の一例を示す図。

【図８】図１に示すアプリケーション開発システムで出力される論理的設計情報の一例を示す図。

【図９】図１に示すアプリケーション開発システムで用いられる物理部品情報（ソフトウェア部品ごとのコード生成方法に関連する情報）の一例を示す図。

【図１０】図１に示すアプリケーション開発システムで用いられる物理部品情報（全体のコード生成方法に関連する情報）の一例を示す図。

【図１１】図１に示すアプリケーション開発システムで生成されるソースコードの一例を示す図。

【図１２】本発明によるアプリケーション開発システムの他の実施の形態を示すシステム構成図。

【図１３】図１２に示すソース生成処理部の動作を説明するためのフローチャート。

【図１４】図１２に示す部品オブジェクト生成処理部の動作を説明するためのフローチャート。

【図１５】図１２に示すアプリケーション開発システムで用いられる物理部品情報（ソフトウェア部品ごとのコード生成方法に関連する情報）の一例を示す図。

【図１６】図１２に示すアプリケーション開発システムで用いられる物理部品情報（全体のコード生成方法に関連する情報）の一例を示す図。

【図１７】図１２に示すアプリケーション開発システムで生成されるソースコードの一例を示す図。

【図１８】本発明によるアプリケーション開発システムが適用される分散開発環境の一例を示す図。

【図１９】従来のアプリケーション開発方法を模式的に説明するための図。

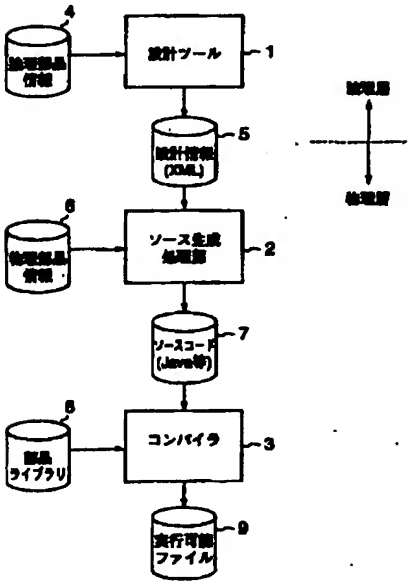
【図２０】本発明によるアプリケーション開発方法を模式的に説明するための図。

【符号の説明】

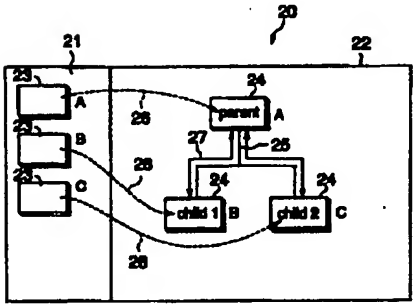
１　設計ツール（論理設計部）
２，２′　ソース生成処理部（物理実装部）
３　コンパイラ（物理実装部）
４　論理部品情報
４ａ　論理部品基本情報
４ｂ　論理部品関連情報
５　論理的設計情報
５ａ　部品属性情報
５ｂ　部品接続情報
６　物理部品情報
７　ソースコード
７ａ　初期設定コード
７ｂ　属性設定コード
７ｃ　部品接続設定コード
７ｄ　終了処理設定コード
８　部品ライブラリ
９　実行可能ファイル
１１　論理部品ツール情報設定部（仕様情報生成部）
１２　論理部品ツール情報（各論理部品の仕様情報）
１２ａ　論理部品入出力仕様
１２ｂ　論理部品接続仕様

12c　論理部品属性仕様
13　設計ツール本体（設計部本体）
14　画面入出力インタフェース部
15　実行処理部
16　部品オブジェクト生成処理部
20　ツール画面
21　パレット領域
22　作業領域
23，24　アイコン
25　関係線
31　クライアントコンピュータ
32　ネットワーク
33　サーバコンピュータ
34　フレキシブルディスク（記録媒体）
35　CD-ROM（記録媒体）
40　アプリケーション
41　プラットフォーム（システム環境）
42　物理部品
43　物理的な組み合わせ情報
52　論理部品
53　論理的な組み合わせ情報

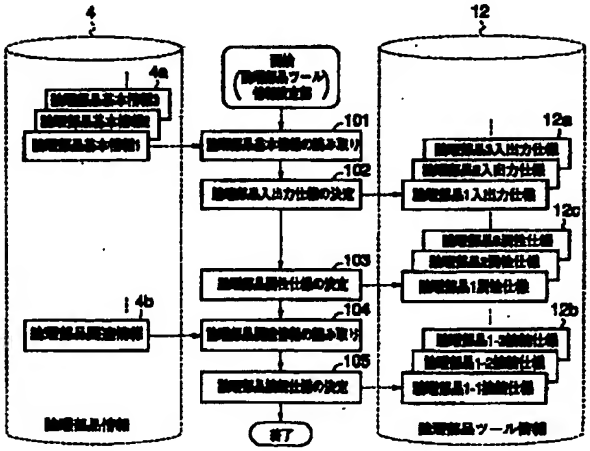
【図1】



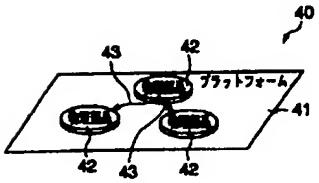
【図3】



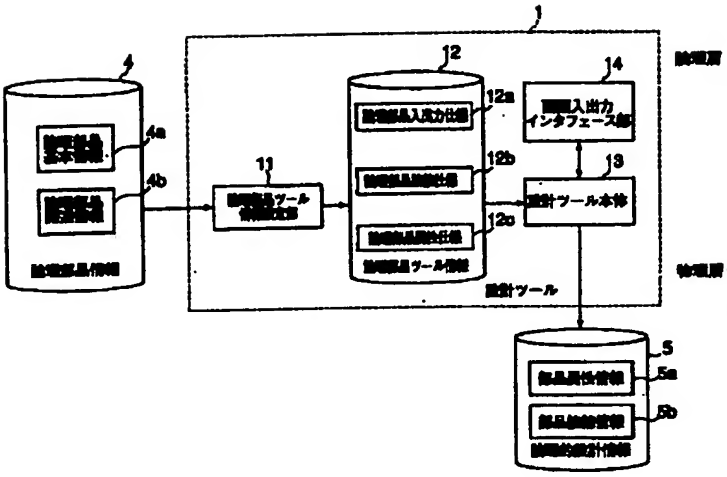
【図4】



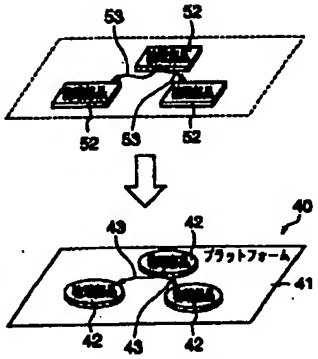
【図19】



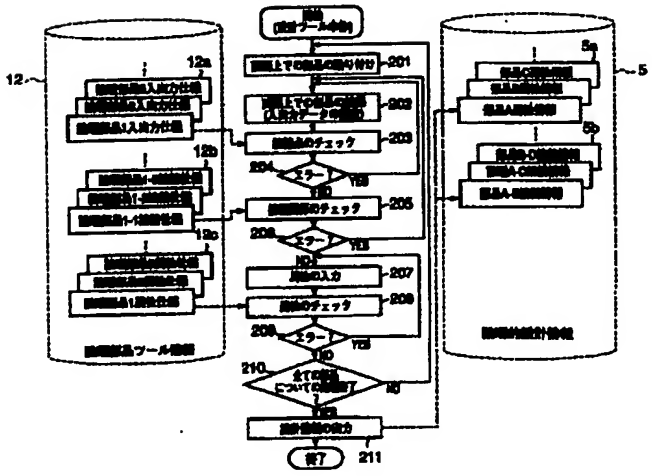
【図2】


1
4
4a
4b
12
12a
12b
12c
14
13
11
設計ツール本体
設計ツール
5
5a
5b

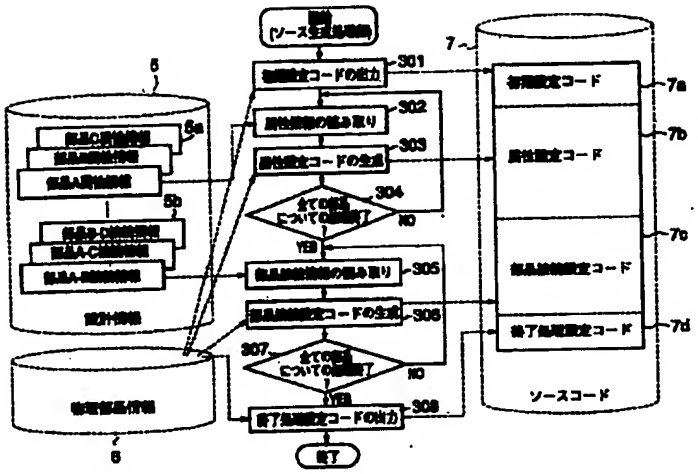
【図20】


53
52
52
53
52
40
43
42
プラットフォーム
41
42
43
42

【図5】


12
12a
12b
12c
5
5a
5b
201
202
203
204
エラー
205
206
207
208
209
210
211
終了

【図6】



【図8】

```
<component type="A",name="parent" option=1>
</component>
<component type="B",name="child1" option=11>
</component>
<component type="C",name="child2" option=12>
</component>
<connection from="parent" to="child1" />
<connection from="parent" to="child2" />
```

Ⓔ Ⓕ

【図15】

```
<componentimpl name="A" impleclass="ClassX">
        <property name="option" type=string"/>
</componenttype>
<componentimpl name="B" impleclass="classY">
        <property name="option" type=string"/>
</componenttype>
<componentimpl name="C" impleclass="classZ">
        <property name="option" type=string"/>
</componenttype>
```

【図7】

(a)

```
<framework name="fw1">
        <component type="A"/>  ···Ⓐ
        <component type="B"/>  ···Ⓐ
        <component type="C"/>  ···Ⓐ
        <connection from="A" to="B"/>
        <connection from="A" to="C"/>   } Ⓑ
</framework>
```

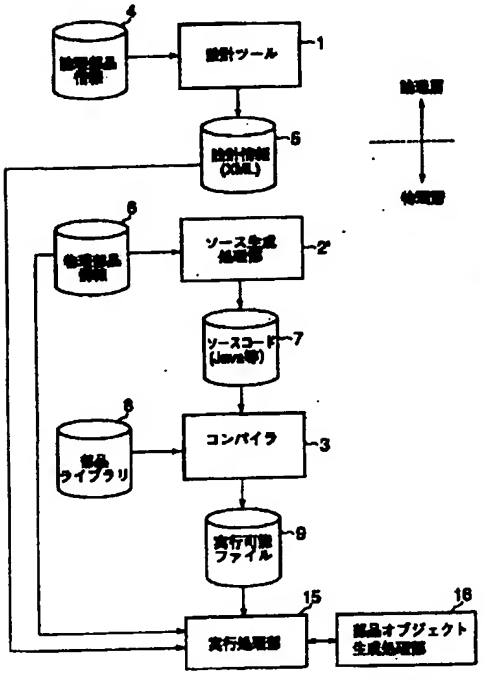
(b)

```
<componenttype name="A">
        <property name="name"type="string" />          } Ⓒ
        <property name="option"type="string" />
        <terminal id="out" type="out" min=1 max=3>     } Ⓓ
                <connectwith type="B"/>
                <connectwith type="C"/>
        </terminal>
</componenttype>
<componenttype name="B">
        <property name="name"type="string" />
        <property name="option"type="string" />
        <terminal id="in" type="in" min=1 max=1>
                <connectwith type="A"/>
        </terminal>
</componenttype>
<componenttype name="C">
        <property name="name"type="string" />
        <property name="option"type="string" />
        <terminal id="in" type="in" min=1 max=1>
                <connectwith type="A"/>
        </terminal>
</componenttype>
```

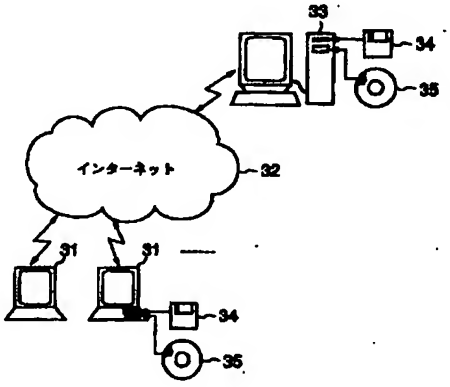
【図9】

```
<componentimpl name="A" implclass="ClassX">
        <new code="ClassX%name%=ClassX.newinstance()" />                    ⑥
        <property name="option"code="%name%.additem("OPTION","%value%","string")" />  ⑧
</componenttype>
<componentimpl name="B" implclass="classY">
        <new code="ClassY%name%=ClassY.newinstance()" />
        <property name="option"code="%name%.additem("OPTION","%value%","string")" />
</componenttype>
<componentimpl name="C" implclass="classZ">
        <new code="ClassZ%name%=ClassZ.newinstance()" />
        <property name="option"code="%name%.additem("OPTION","%value%","string")" />
</componenttype>
<connectionimple>
        <code="%from%.additem("CHILD",%to%); />
        <code="%to%.additem("PARENT",%from%); />                            } ⑦
</componenttype>
```

【図12】



【図18】



【図10】

(a)

```
<frameworkimpl name="fw1" >
        <init file="fw1init.java" /> --------- ⑨
        <term file="fw1term.java" /> ------- ⑩
</componenttype>
```

fw1 init.java

(b)

```
import java. lang. *;
import java. util. *;
    ・
    ・(各種定義項目)
    ・

public class AA1 implements AA
{
        public Object bbbb()
        {
          ・
          ・(各種初期化処理)
          ・
```

fw1 term.java

(c)

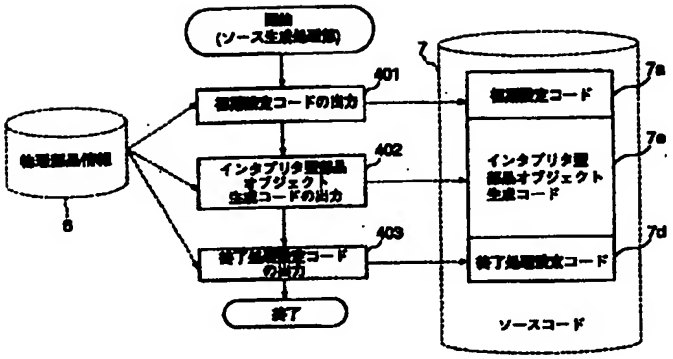
```
          ・
          ・(各種終了処理)
          ・
        }
}
```

【図11】

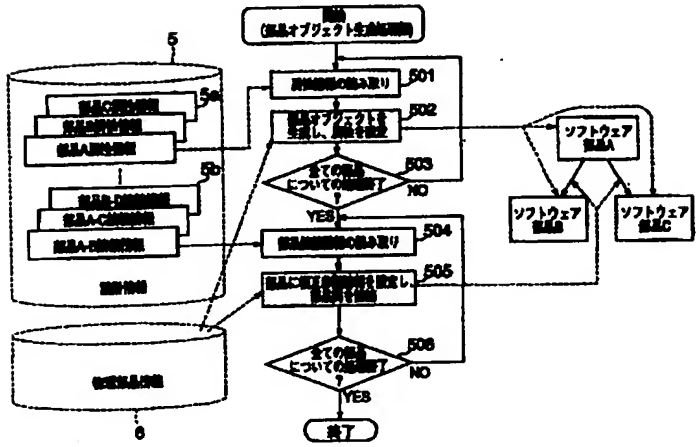
```
import java. lang. *;
import java. util. *;
    *
    * (各種定義項目)
    *

public class AA1 implements AA
{
        public Object bbbb()
        {
          *
          * (各種初期化処理)
          *
        ClassX parent=ClassX.newinstance();
        parent.addItem("OPTION","1","string")";
        ClassY child1=ClassY.newinstance();
        child1.addItem("OPTION","11","string")";
        ClassZ parent=ClassZ.newinstance();
        parent.addItem("OPTION","12","string")";
        parent.addItem("CHILD",child1);
        child1.addItem("PARENT",parent);
        parent.addItem("CHILD",child2);
        child2.addItem("PARENT",parent);
          *
          * (各種終了処理)
          *
        }
}
```

【図13】


開始
(ソース生成処理部)
401
402
403
終了
7
7a
7c
7d
インタプリタ型
部品オブジェクト
生成コード
ソースコード
6

【図14】


開始
(部品オブジェクト生成処理部)
5
5a
5b
501
502
503
YES
NO
504
505
506
終了
ソフトウェア
部品A
ソフトウェア
部品B
ソフトウェア
部品C
6

【図16】

(a)

```
<frameworkimpl name="fw1">
        <init file="fw1init.java"/> -------- (L)
        <main file="fw1main.java"/>---- (M)
        <term file="fw1term.java"/>------(N)
</componenttype>
```

fw1 init.java

(b)

```
import java. lang. *;
import java. util. *;
    ・
    ・(各種定義項目)
    ・

public class AA1 implements AA
{
        public Object bbbb()
        {
          ・
          ・(各種初期化処理)
          ・
```

fw1main.java

(c)

```
          ・
          ・
          ・
          ・(インタプリタ型部品オブジェクト生成処理)
          ・
          ・
          ・
```

fw1term.java

(d)

```
          ・
          ・(各種終了処理)
          ・
        }
}
```

【図17】

```
import java.lang.*;
import java.util.*;
    ・
    ・(各種定義項目)
    ・

public class AA1 implements AA
{
        public Object bbbb()
        {
          ・
          ・(各種初期化処理)
          ・
          ・
          ・
          ・(インタプリタ型部品オブジェクト
                        生成処理)
          ・
          ・
          ・
          ・
          ・
          ・(各種終了処理)
          ・
        }
}
```